A Flexible Extension of XPath to improve XML Querying

Ernesto Damiani
University of Milan
via Bramante 65
26013 - Crema (CR) Italy
damiani@dti.unimi.it

Stefania Marrara
University of Milan
via Bramante 65
26013 - Crema (CR) Italy
marrara@dti.unimi.it

Gabriella Pasi
University of Milan Bicocca
viale Sarca 336
20126 - Milan Italy
pasi@disco.unimib.it

ABSTRACT

This work presents a flexible XML selection language, *FleX-Path* which allows the formulation of flexible constraints on both structure and content of XML documents. Some experimental results, obtained with a preliminary prototype, are described in order to show that the idea promises good results.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.3 Information Search and Retrieval—*Query Formulation*

General Terms

Design, Languages

Keywords

XML retrieval, XPath, flexible query, Fuzzy Set Theory

1. INTRODUCTION

In the last few years, the problem of modeling and querying XML documents in a flexible way has been intensively studied [6, 8, 7]. In particular, several works in the literature have addressed the problem of defining IR models for semi-structured documents [4, 9].

The W3C has defined the language XPath for selecting XML node sets via *tree traversal* expressions. XPath selection is Boolean in nature: it partitions XML nodes into those which fully satisfy the selection condition, and those which do not. However, Boolean conditions are —in some scenarios— not suitable for effectively querying XML documents. To justify this claim we note that even when XML schemas do exist, they may be not available to users. Moreover document trees with the same schema may be very different (both in used tags and nesting), and hence the schema will allow for diverse instantiations, making it difficult to predict a particular document structure (from the schema). Finally, the same XML tree can be sometimes described using different schemas. As a consequence, users often end up defining *blind* queries, i.e. queries written without a precise knowledge of the schema. In these cases the availability of a flexible query language allowing for approximate queries would be extremely helpful.

This work focuses on an extension of XPath to the aim of allowing the specification of flexible constraints on both textual content (in the IR style) and document structure. A first contribution in this direction was presented in [7]. In the present research activity the following problems are addressed: (i) definition of flexible constraints on the content of documents, (ii) definition of flexible constraints on the structure of documents, in order to find close matches to structural query conditions; (iii) focused search for retrieving only the most relevant parts of documents.

One of the most important outcomes of this research is that the produced result of a FleXPath query evaluation is a ranked list of fragments, while the XPath query evaluation produces a set of results.

Copyright is held by the author/owner(s).
SIGIR'08, July 20–24, 2008, Singapore.
ACM 978-1-60558-164-4/08/07.

2. DESCRIPTION OF THE APPROACH

In [5] a first proposal of the approach was presented, based on the idea of using Fuzzy Set Theory to introduce flexibility in XPath through the definition of some flexible constraints. Based on that preliminary work, in this paper two new classes of flexible constraints are introduced:

- *Flexible constraints on node content and XML tags*: to the aim of specifying flexible selection conditions on node contents and tag names, (e.g, `SIMILAR`, `AROUND` and `CLOSE`);

- *Fuzzy Subtree Matching constraints*, namely `NEAR`, `BELOW`, and `APPROXIMATELY`.

From a formal point of view the XPath syntax is extended to allow the expression of flexible constraints. On the side of the evaluation strategy of a FleXPath query, the approach is based on a two steps process. The first step performs a pre-processing of the query finalized at evaluating the content based constraints. This is performed by an usual inverted file approach. This first evaluation step produces a ranked list of documents, which constitutes the input to the second phase. The aim of the second step is to evaluate, on the pre-selected documents, the flexible structural constraints.

The retrieved results are shown in a bi-dimensional space in which one dimension is the content-based evaluation retrieval status value and the other is the structure-based retrieval status value. The user will select the dimension he/she prefers to rank the results in the list of retrieved items. At the present stage of our research the two numeric values are not combined (as they have a different semantics).

The proposed approach can be implemented in two distinct ways. Either a full flexible XPath engine is implemented, or it is possible to construct the evaluation functions of the flexible constraints as a separate module outside

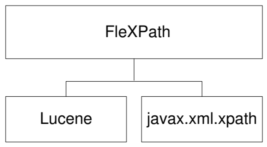

Figure 1: Architecture of the FleXPath prototype

any commercial XPath query engine. The second incremental approach (see Figure 1) has been followed and the first prototype was developed as a top layer of the open source software *Lucene* [1] and the XPath engine *Apache javax* [2], leaving the development of a full FleXPath engine to the next future.

3. FLEXIBLE CONSTRAINTS ON CONTENT AND STRUCTURE

This section introduces the flexible constraints that the user can specify in a query using FleXPath. Informally, `AROUND` can be used to specify a constraint tolerant to the mismatch of numbers and dates, `CLOSE` allows to express a string similarity among different string values, while `SIMILAR` allows to specify in the query a selection of nodes with a tag name similar to a given one.

On the structure side, the constraint `BELOW`, inserted into the axis of a path expression, allows to extract elements, attributes or text that are direct descendants of the current node, giving a penalty which is proportional to the result's distance from the current node. The constraint `NEAR` (see example in Section 4) represents its generalization. In this case the similarity evaluation is performed within any axis. The last constraint, `APPROXIMATELY`, is related to fan-out and allows to select the elements with a given name that have a number of direct descendants close to the one indicated in the query. It is a derived constraint that can be substituted by a `BELOW` constraint applied to the result of the COUNT operator on the children of a given node. The constraints are either defined as fuzzy subsets on the appropriate reference domain or by similarity measures.

3.1 Query evaluation

This section describes the execution plan of a query expressed using FleXPath. Suppose that a user proposes a tree-pattern query containing flexible constraints both in the content and in the structure. The query engine will evaluate this query in four steps: I. initially the query is divided into its two components: content and structure constraints; II. the engine evaluates the content constraints and produces a ranked list of possible results; III. then the structure is evaluated. A flexible constraint is mapped to a sequence of new XPath queries, each one characterized by its distance from the original query, that simulate the behavior of the flexible constraint of the user's query. IV. All new queries are executed and the results are shown in a bi-dimensional space with structure rank on the x-axis and content rank on the y-axis.

Obviously each degree of flexibility introduced in the query will be paid by increasing the number of queries computed by the engine in order to simulate the flexible query. However this drawback is bounded by the pre-selection step that reduces the set of documents target of the execution of the final set of queries.

The user can select the dimension he/she prefers to rank the results in a ordered list.

4. PRELIMINARY RESULTS

To the aim of showing that the idea is promising and worthy of further study and development efforts, some experiments have been performed using XMark [3]. A set of 17 flexible queries, containing one or more flexible constraints, combined in different ways, has been evaluated on a collection of 2556 documents in order to compare the behavior of FleXPath w.r.t. XPath. An example of FleXPath query is:

```
site/regions{NEAR}description[//listitem/
          text(){CLOSE}'blueberry']//text
```

The user is interested in a `text` node, descendant of a node `description`. The user only knows that the target `description` should be structurally *near* (but not strictly descendant of) the node `regions` and its sibling `listitem` content contains a word similar to (*close*) blueberry. Preliminary results show that the recall of FleXPath is, as expected, generally higher than the recall of XPath.

A first direct result of the use of FleXPath is that several queries, that obtained empty results using XPath, once reformulated in a more flexible way with FleXPath produced useful results.

5. CONCLUSIONS

This paper proposes an extension of the syntax of XPath by means of flexible constraints. Some preliminary experiments show that FleXPath allows the formulation of queries able to capture relevant information when the user does not know the exact structure of the dataset or the exact words that represent the desired content. Future work will include an extension of this idea to XQuery and the development of a new flexible query engine for XML documents to be tested on an INEX collection.

6. REFERENCES

[1] Apache lucene. http://lucene.apache.org/java/docs/.

[2] javax xpath engine, http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/xpath/package-summary.html.

[3] XMark: An XML benchmark project. http://monetdb.cwi.nl/xml/.

[4] G. Bordogna and G. Pasi. Personalized indexing and retrieval of heterogeneous structured documents. *Information Retrieval*, 8(2):301–318, 2005.

[5] D. Braga, A. Campi, E. Damiani, G. Pasi, and P. Lanzi. FXPath: Flexible querying of XML documents. In *Proceedings of EuroFuse 2002, Varenna, Italy, Sep.*, 2002.

[6] E. Damiani and L. Tanca. Blind queries to XML data. In *Database and Expert Systems Applications*, pages 345–356, 2000.

[7] N. Fuhr and K. Grobjohann. XIRQL: A query language for information retrieval in XML documents. In ACM, editor, *Proceedings of SIGIR'01, New Orleans, Luisiana, USA*, 2001.

[8] H.-G. Li, S. A. Aghili, D. Agrawal, and A. E. Abbadi. FLUX: Fuzzy content and structure matching of XML range queries. In *Proceedings of WWW 2006, May 23-26, 2006, Edinburgh, Scotland*, 2006.

[9] R. Wilkinson. Effective retrieval of structured documents. In *Proceedings of the 17th ACM-SIGIR, Dublin.*, pages 311–317, 1994.